Listas e Programação Funcional

Maria Madrugo

Introdução

O objectivo deste texto é mostrar que há muita coisa que se pode fazer no Mathematica apenas usando operações de listas e programação funcional, sem recorrer a indexação excessiva nem iteradores. É suposto ser razoavelmente elementar, e fica muita coisa ainda por explorar. Por exemplo, não irei falar de comandos como NestWhile nem FixedPoint. Se quiseres aprender mais, aqui estão links para a documentação do Mathematica, nomeadamente para:

- O tutorial de Álgebra Linear
- O guia de Programação Funcional

Criar listas e aceder a elementos

A forma mais fácil de definir listas é dizendo que elementos tem, entre chavetas e separados por vírgulas. Por exemplo, a é uma lista:

a={1,2,4,2}

Da mesma forma, podemos definir matrizes, que são listas de listas. Por exemplo, b é uma matriz 2x2, com a primeira linha {1,2} e a segunda {5,4}:

b={{1,2},{5,4}}

Existem comandos que definem certos tipos de matrizes e listas, como por exemplo ConstantArray, IdentityMatrix e DiagonalMatrix. Além disso, o comando Table permite definir matrizes ou vectores definindo uma expressão para cada entrada, por exemplo:

```
Table[Prime[i],{i,2,10}]
Table[{i,j},{i,-1,1},{j,-1,1}]
```

No primeiro caso, definimos a lista {3,5,7,11,13,17,19,23,29}, uma vez que esta tem na entrada i o i-ésimo primo (Prime[i]), com o i a ir de 2 até 10. No segundo, definimos a lista {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}}, desta vez com dois indíces, i e j, a ir de -1 a 1.

Para aceder a elementos duma lista, usam-se parentesis rectos e começa-se a contar do 1. Por exemplo, a[[1]] é 1 e a[[3]] é 4. Em matrizes, funciona da mesma forma, por exemplo b[[1]][[1]] ou b[[1,1]] é 1 e b[[2]][[1]] ou b[[2,1]] é 5. É possível cortar listas e matrizes usando o comando Span(;;), que poderás ver como funciona na documentação.

Operações Básicas

Dadas operações básicas, o Mathematica tende a conseguir fazê-las componente a componente com facilidade. Geralmente, isto manifesta-se numa de duas situações:

- Operações entre listas do mesmo tamanho, como:
  - {1,2,3}+{1,4,2}, que retorna {2,6,5};
  - {3,2,1}^{2,3,10}, que retorna {9,8,1};

  - {{1,2},{2,3}}*{{2,1},{4,2}}, que retorna {{2,2},{8,6}}. [1]

- Operações entre uma lista e um escalar:
  - {1,2,3}-2, que retorna {-1,0,1};
  - {{2,4},{1,3}}*3, que retorna {{6,12},{3,9}}.

Por vezes, pode dar jeito fazer uma mistura destas duas. Por exemplo, dado {{1, 2}, {3, 4}}*{2, -1}, como as listas têm o mesmo tamanho (apesar de uma ser lista de listas e outra lista de escalares), o Mathematica multiplica {1,2} por 2 e {3,4} por -1 (cada um dos quais entra no segundo caso), e retorna {{2, 4}, {-3, -4}}.

Há também muitas outras operações, chamadas *Listable*, que podem receber uma lista e são aplicadas a cada componente, como Sin, Cos, Prime e Log.

Finalmente, dada uma lista, existem três quantidades fáceis de calcular:

- Length[list] retorna o tamanho duma lista. No caso duma matriz retorna a quantidade de linhas;
- Dimensions[list] retorna as dimensões de uma lista/matriz;
- Total[list] retorna a soma dos elementos da lista;

Aplicar funções a listas

Comando Map

Dada uma função, é comum querermos aplicá-la a listas. Na secção anterior, vimos alguns exemplos de funções listáveis, que são automaticamente aplicadas componente a componente. No entanto, nem sempre é esse o caso, e existe um comando que nos permite 'obrigar' qualquer função a ser aplicada componente a componente, o Map:

```
a={{1,2,3},{2,1,3,4}}
Length[a]
Map[Length,a]
```

Como esperado, a segunda linha retorna 2, uma vez que a lista a tem dois elementos, ambos listas. Na linha seguinte, o Map aplica o comando Length a cada uma destas listas, retornando {3,4}, o número de elementos de cada.

Comando Apply

Apesar de ligeiramente menos comum, existe outra forma natural de querer aplicar funções a listas: considerar os elementos como os argumentos nas listas. Existe um exemplo de aplicação simples, que é fazer o produto de todos os elementos. Para isso, podemos tentar usar o facto da função Times (tipicamente usada na forma binária *) poder receber mais de dois elementos duma vez. No entanto, Times[list] retorna apenas a lista, porque considera que apenas tem um argumento. O comando Apply resolve esse problema:

```
Apply[Times,{1,2,3,4}]
```

Tal como gostaríamos, esta instrução retorna 24, e é bastante mais simples do que usar While, Do, For, ou até Product, uma vez que não requer indexação.

Comando Select

Um tipo especial de funções que podemos querer aplicar a listas são funções boleanas, que retornam um valor de verdade, como por exemplo EvenQ, que retorna True quando um número é par. A forma mais fácil de aplicar esta função a listas é a seguinte:

```
EvenQ[{1,2,3,4}],
```

[1] De notar que isto não corresponde à multiplicação de matrizes, que é efectuada usando o comando Dot, ou apenas um ponto (A.B)

Este comando retorna {False,True,False,True}. Apesar de isto já ser informação útil, por vezes queremos seleccionar os elementos que cumprem essa condição. Para isso, podemos usar o comando Select:

Select[{1,2,3,4},EvenQ]

Tal como desejado, com esta rotina obtemos {2,4}. O Select também funciona com funções feitas à mão, por exemplo

Select[{11,4,1,19},Function[x,x>10]].

Este comando devolve os elementos da lista maiores que 10, ou seja, {11,19}

Truques úteis

Comando Nothing

Às vezes no Table queremos usar If, de forma a que no caso negativo não queiramos adicionar nada à lista. Nesse caso, podemos adicionar Nothing, por exemplo:

Table[If[PrimeQ[i],Nothing,i],{i,1,10}]

Este comando retorna os números até 10 que não são primos, nomeadamente {1,4,6,8,9,10}.

Comando Flatten

O Table cria uma dimensão para cada índice. Por exemplo,

list=Table[{a,b,c},{a,0,1},{b,0,1},{c,0,1}]

Este comando cria uma lista com 4 dimensões, uma para a, uma para b e uma para c e uma em a,b,c. Em particular, o comando Dimensions[list] retorna de facto {2,2,2,3}. Muitas vezes dava-nos jeito obter uma lista de trios a,b,c, em vez de ter uma estrutura tão complicada. Um motivo simples é o uso de Select, que itera nos elementos duma lista, apenas andando na primeira dimensão.

Para obter uma lista com duas dimensões (uma delas em {a,b,c}) a partir de list, podemos usar o seguinte comando:

Flatten[list,2],

onde o 2 indica que queremos descer 2 níveis, 2 dimensões, 2 chavetas. Para ver melhor esta questão dos níveis, seja newlist={{{a,b},{c,d}},{{e,f},{g,h}}}. Então,

Flatten[newlist]={a,b,c,d,e,f,g,h}
Flatten[newlist,1]={{a,b},{c,d},{e,f},{g,h}}

Comando Transpose

O Transpose[2] é frequentemente útil, um dos casos comuns sendo quando queremos passar de pares de listas para listas de pares. Por exemplo, se quisermos pares {i,Prime[i]}, uma alternativa a usar Table seria:

Transpose[Range[i],Prime[Range[i]]].

Um exemplo de aplicação é preparar dados para fazer o plot usando ListPlot.

[2] Quando recebe uma matriz, este comando corresponde a fazer a transposta no sentido de Álgebra Linear.